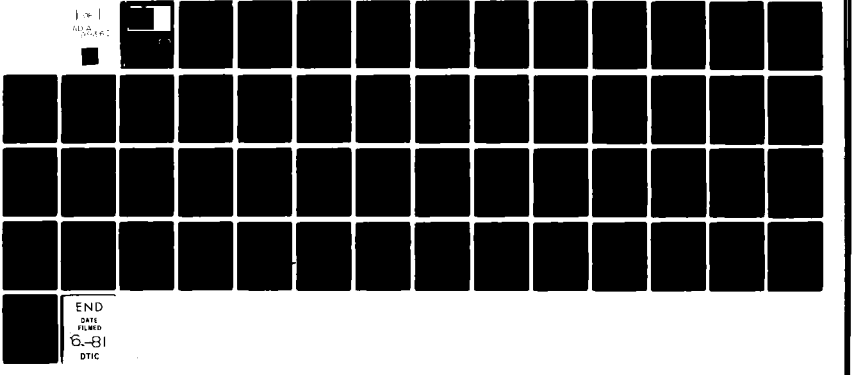
AD-A099 362 WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F/G 12/1
A DECOMPOSITION METHOD AND ITS APPLICATION TO BLOCK ANGULAR LIN--ETC(U)
JAN 81 C D HA DAAG29-80-C-0041
UNCLASSIFIED MRC-TSR-2174 NL



END
DATE FILMED
6-81
DTIC

LEVEL II

AD A099362

MRC Technical Summary Report #2174

A DECOMPOSITION METHOD AND ITS APPLICATION TO BLOCK ANGULAR LINEAR PROGRAMS

Cu Duong Ha

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53706

January 1981

Received November 7, 1980

Approved for public release
Distribution unlimited

Sponsored by
U. S. Army Research Office
P.O. Box 12211
Research Triangle Park
North Carolina 27709

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

A DECOMPOSITION METHOD AND ITS APPLICATION TO BLOCK ANGULAR LINEAR PROGRAMS

Cu Duong Ha†

Technical Summary Report #2174
January 1981

ABSTRACT

In this paper we propose and develop techniques for solving structured, large-scale convex programming problems. The procedure is a combination of a decomposition technique of Dantzig-Wolfe type and the proximal point method. The proximal point method is used to overcome the drawbacks of the decomposition technique.

The procedure is then used to solve block angular linear programming problems. By exploiting the linearity of the problem we have several variants of the procedure.

AMS(MOS) Subject Classifications: 90C06, 90C25

Key Words: Decomposition method, convex programming, large scale systems, linear programming

Work Unit Number 5 - Operations Research

†Current address: Mathematical Sciences Department
Virginia Commonwealth University
Richmond, VA 23284

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

SIGNIFICANCE AND EXPLANATION

Optimization problems which arise from real-world situations are usually complex and large. It is almost impossible to use general methods to solve these problems. Fortunately, these problems always have certain special features, and by exploiting these special features we can obtain an optimal solution for the problem. The decomposition method is one of the solution techniques that make use of the special structure of the problem. The idea of the decomposition method is to break up a problem into a sequence of simpler subproblems, and then obtain the optimal solution for the problem from the optimal solutions of the subproblems.

In this paper we discuss a decomposition technique and its disadvantages, and then propose to use the proximal point method to overcome these disadvantages. We also apply the proposed procedure to solve block angular linear programming problems. An important application of block angular linear programming is the problem of a multidivisional organization: The divisions of the organization operate almost independently of each other; they are only coupled by the fact that they share a few scarce resources. The decomposition method seems to be an appropriate approach for solving this type of problem.

A DECOMPOSITION METHOD AND ITS APPLICATION TO BLOCK ANGULAR LINEAR PROGRAMS

CU DUONG HA[†]

1. Introduction.

A general mathematical programming problem with more than a few hundred variables is not easy to solve, especially in nonlinear cases. Fortunately, large problems, which arise from practical applications, nearly always have some special features; for example, sparseness of the matrix of constraints or block angular structure of the problem. In general, it is only by exploiting these special structures that we can solve large-scale problems.

The idea of decomposing a problem into a collection of simpler problems is not new in mathematics. However, only in recent years have decomposition techniques been extensively developed for mathematical programming, primarily to take advantage of powerful computer systems. Decomposition approaches are essentially suitable for systems composed of several smaller subsystems which are independent of each other except for a few weak connections between them. It seems natural, then, to break up these weak interconnections and handle each subsystem independently, obtaining the solution for the overall system from the solutions of the subsystem.

[†]Current address: Mathematical Sciences Department
Virginia Commonwealth University
Richmond, VA 23284

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

In Section 2 we discuss the underlying idea of a decomposition technique of Dantzig-Wolfe type applied to problems of the form

$$\inf \{ \sum_{i=1}^{m} f_i(x_i) \mid \sum_{i=1}^{m} A_i x_i = a \} ,$$

where, for fixed i, x_i is a vector, A_i is a matrix, and f_i is a convex function having values in the extended real line $(-\infty,+\infty]$. This decomposition technique is very interesting but it has several drawbacks in some important cases. To overcome these drawbacks we combine the proximal point method with the decomposition technique and propose a new decomposition procedure to solve convex, structured large-scale problems.

Section 3 deals with the application of the general procedure in Section 2 to block angular linear programming problems. An example of this type of problem is the problem of a multidivisional organization. Each division is operated independently from the others; the only linkage between them is that they all share a few scarce resources. By exploiting the linearity of the constraints we can modify the general algorithm and have specific algorithms for block angular linear programming problems. Computational aspects of the algorithms are reported; they show that, for large problems, our procedure is better than some other methods.

2. Decomposition method

In this section we shall present a new approach to solved structured convex programming problems. This approach is a combination of a decomposition technique of Dantzig-Wolfe type and the proximal point method.

2.1 The decomposition technique of Dantzig-Wolfe type.

We consider the problem

$$\inf \{ \sum_{i=1}^{m} f_i(x_i) \mid \sum_{i=1}^{m} A_i x_i = a \} \tag{2.1}$$

where $x_i \in \mathbb{R}^{n_i}$, f_i is a closed proper convex function from $\mathbb{R}^{n_i}$ to $(-\infty,+\infty]$, A_i is $\ell \times n_i$ matrix, and $a \in \mathbb{R}^{\ell}$. Let $n = \sum_{i=1}^{m} n_i$ and $x := (x_1, x_2, \ldots, x_m)$.

Note that we allow f_i to have the value $+\infty$. In that way we can incorporate constraints into the objective function.

First we need several definitions. Let f be a convex function defined on a subset S of $\mathbb{R}^n$ and having values on $[-\infty,+\infty]$. The set

$$\{(x,\mu) \mid x \in S,\ \mu \in \mathbb{R},\ \mu \geq f(x)\}$$

is called the epigraph of f and is denoted by epi f. The effective domain of f, denoted by dom f, is defined by

$$\text{dom } f := \{x \mid \exists \mu,\ (x,\mu) \in \text{ epi } f\} ;$$

we can easily see that

$$\text{dom } f = \{x \mid f(x) < \infty\} .$$

The conjugate function f^* of f is defined by

$$f^*(p) := \sup_x \{\langle p,x\rangle - f(x)\}$$

where $\langle p,x \rangle$ is the inner product of vectors $p \in \mathbf{R}^n$ and $x \in \mathbf{R}^n$.

A convex function is said to be closed if its epigraph is a closed set in $\mathbf{R}^{n+1}$.

A convex function f is said to be proper if $f(x) < \infty$ for at least one x and $f(x) > -\infty$ for every x.

It is easy to show that the conjugate function f^* of any convex function f is a closed convex function and f^* is proper if and only if f is proper. Now we return to our problem (2.1). The linear constraints

$$A_1x_1 + A_2x_2 + \ldots + A_mx_m = a$$

are called coupling constraints. They interconnect m subproblems; each has its own set of variables and constraints. Without the coupling constraints, the problem (2.1) could be solved easily by solving separately m small subproblems $\inf f_i(x_i)$.

The decomposition techniques for (2.1) are methods used to break up the coupling constraints, so that the resulting problem can be separated into smaller subproblems. One way to achieve that purpose is to use the following well-known results of convex analysis.

Theorem 2.1

Under the conditions

$$\text{im } A_i^T \cap \text{ri dom } f_i^* \neq \phi \qquad i = 1,2,\ldots,m \tag{2.2}$$

where im A_i^T is the image of $\mathbb{R}^\ell$ under A_i^T and ri dom f_i^* is the interior of the effective domain of f_i^*,

the infimum in (2.1) is equal to

$$\sup_y \{\langle a,y\rangle - \sum_{i=1}^m f_i^*(A_i^T y)\} . \tag{2.3}$$

It is attained for some $\bar{x} = (\bar{x}_1,\ldots,\bar{x}_m)$ if the problem (2.1) is feasible and is $+\infty$ if (2.1) is infeasible.

(For a proof see [Rockafellar, 1970, page 142].)

Define

$$g(y) := \langle a,y\rangle - \sum_{i=1}^m f_i^*(A_i^T y) . \tag{2.4}$$

Then (2.3) can be rewritten as

$$\sup_y g(y) . \tag{2.5}$$

Note that the dimension ℓ of y is usually much smaller than n, the number of variables of (2.1). Given a value of y, we have to compute

$$f_i^*(A_i^T y) = \sup_{x_i} \{\langle A_i^T y, x_i\rangle - f_i(x_i)\} \tag{2.6}$$

for $i = 1,2,\ldots,m$; which are m small subproblems. The scheme to solve (2.1) using the above results is: choose a value of y and solve m subproblems (2.6) corresponding to that fixed y. If y is an optimal solution for (2.5), stop. Otherwise update y and solve (2.6) again. Repeat until an optimal solution of (2.5) is obtained.

Using this scheme, we have successfully decomposed the problem (2.1) with n $(= \sum_{i=1}^{m} n_i)$ variables into a sequence of small subproblems having $n_i (i=1,\ldots,m)$ and ℓ variables. Because of the following interpretations, the dual problem (2.5) is usually called the master program and the decomposition technique above, the price-directive approach. Set prices (values of y) on the common resources (coupling constraints) and add the costs of these resources to the objective function of each subproblem (terms $\langle A_i^T y, x_i \rangle$ in (2.6)). By appropriately varying these prices, one can cause the subproblems to produce solutions which yield an optimal solution for the original problem. A diagram for the scheme is shown below.

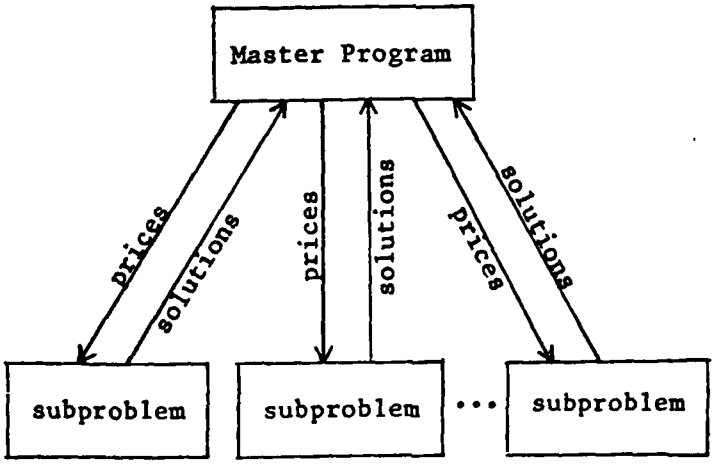


Figure 2.1 A decomposition scheme for large-scale systems.

Now, let us examine the conditions (2.2) to see if they are reasonable. First, if f_i^* are polyhedral (i.e. the epigraphs of f_i^* are polyhedral convex sets), then the conditions (2.2) can be weakened to

$$(\text{im } A_i^T) \cap (\text{dom } f_i^*) \neq \phi \quad i = 1,2,\ldots,m \ .$$

Second, consider the general linear programming problem

$$\begin{aligned} &\min \ \langle c,x\rangle \\ &\text{s.t.} \ \ Ax = a \\ &\qquad x \geq 0 \ . \end{aligned}$$

Its dual problem is

$$\begin{aligned} &\max \ \langle a,y\rangle \\ &\text{s.t.} \ \ A^T y \leq c \ . \end{aligned}$$

The function f in this case is defined by

$$f(x) := \begin{cases} \langle c,x\rangle & \text{if } x \geq 0 \\ +\infty & \text{otherwise,} \end{cases}$$

then the conjugate function f^* of f is

$$\begin{aligned} f^*(A^T y) &= \sup \{\langle A^T y,x\rangle - f(x)\} \\ &= \begin{cases} 0 & \text{if } A^T y \leq c \\ +\infty & \text{otherwise .} \end{cases} \end{aligned}$$

Thus the condition (2.2) in the case of linear programming is the requirement that the feasible region of the dual problem is nonempty. Finally, if $f_i^*(p_i)$ is finite for every p_i in R^{n_i} (it holds in our applications), then (2.2) are satisfied trivially because the effective domain of f_i^* is the entire space R^{n_i}.

The decomposition technique described above is very elegant, but in applications there are several drawbacks:

(i) The function $g(y)$ defined by (2.4) may not be finite for some y. The set $Y := \{y | -\infty < g(y)\}$ may be considered as the feasible region for the problem (2.5). But Y does not have an explicit form because it is defined by m maximization problems (2.6). ($y \in Y$ if and only if $(f_i^*(A_i^T y) < \infty$ for all i)).

(ii) $g(y)$ may not be differentiable everywhere in the set Y. Shapiro has a paper [1978] discussing this point in more detail.

(iii) Given a y, the corresponding subproblems (2.6) may not have unique optimal solutions. Consequently, arbitrary optimal solutions $x_i(y^*)$ of (2.6) corresponding to the optimal solution y^* of (2.5) may not aggregate to form an optimal solution for the original problem (2.1) (although <u>some</u> such solution will do so).

We illustrate those points by a simple example:

$$\begin{aligned} \min \quad & x_1 + 2x_2 \\ \text{subject to} \quad & x_1 \leq 2 \\ & x_2 \leq 3 \\ & x_1 + x_2 = 4 \\ & x_1 \text{ unrestricted}, \quad x_2 \geq 0 . \end{aligned}$$

In this example x_1 and x_2 are real numbers

$$f_1(x_1) = \begin{cases} x_1 & \text{if } x_1 \leq 2 \\ +\infty & \text{otherwise} \end{cases}$$

$$f_2(x_2) = \begin{cases} 2x_2 & \text{if } 0 \leq x_2 \leq 3 \\ +\infty & \text{otherwise} \end{cases}$$

$A_1 = 1$, $A_2 = 1$

$$f_1^*(p_1) = \sup_{x_1} \{\langle p_1, x_1 \rangle - f_1(x_1)\} = \sup_{x_1 \leq 2} (p_1 - 1)x_1$$

$$= \begin{cases} +\infty & \text{if } p_1 < 1 \\ 2(p_1 - 1) & \text{if } p_1 \geq 1 \end{cases}$$

$$f_2^*(p_2) = \sup_{x_2} \{\langle p_2, x_2 \rangle - f_2(x_2)\} = \sup_{0 \leq x_2 \leq 3} (p_2 - 2)x_2$$

$$= \begin{cases} 0 & \text{if } p_2 \leq 2 \\ 3(p_2 - 2) & \text{if } p_2 \geq 2 \ . \end{cases}$$

The images of A_1^T and A_2^T are $\mathbb{R}$, so the conditions (2.2) are satisfied trivially. It is easy to compute g in this case:

$$g(y) = \begin{cases} -\infty & , y < 1 \\ 2y + 2 & , 1 \leq y \leq 2 \\ -y + 8 & , 2 \leq y \ . \end{cases}$$

We can see that (i) if $y < 1$ then $g(y)$ is infinite; (ii) g is not differentiable at the point $y = 2$; (iii) the optimal solution of the problem (2.5) is $y^* = 2$, the corresponding subproblem

$$\sup_{x_1 \leq 2} (y^*-1)x_1$$

has a unique optimal solution $x_1(y^*) = 2$ but the subproblem

$$\sup_{0 \leq x_2 \leq 3} (y^*-2)x_2$$

has an infinite number of optimal solutions. If we use the simplex method to solve it, we have $x_2(y^*) = 0$ or 3. Neither (2,0) nor (2,3) is the optimal solution of the original problem, which is (2,2).

All of the drawbacks above will disappear if the functions f_i are strongly convex.

Definition. A function f defined on a convex $S \subset \mathbb{R}^n$ is said to be strongly convex with modulus $\alpha > 0$ if

$$f((1-\lambda)x+\lambda y) \leq (1-\lambda)f(x) + \lambda f(y) - \tfrac{1}{2}\alpha\lambda(1-\lambda)\|x-y\|^2$$

for all $x,y \in S$ and $0 < \lambda < 1$.

We have the following nice properties in the case that f_i are strongly convex.

Theorem 2.2

If the f_i are strongly convex then the f_i^* are finite everywhere and are Lipschitz continuously differentiable. Hence g is also finite everywhere and is Lipschitz continuously differentiable.

The derivative g' of g at the point y is given by

$$g'(y) = a - \sum_{i=1}^{m} A_i(x_i(y))$$

where $x_i(y)$ is the optimal solution of (2.6) for $i = 1,2,\ldots m$. If y^* is the optimal solution of (2.5) then $x_i(y^*)$ are optimal solutions for the original problem (2.1).
(For a proof see Robinson [1978]).

Remark. In this case, the conditions (2.2) are satisfied trivially because the f_i^* are finite everywhere.

Strong convexity is a very attractive property, but unfortunately, in applications there are several important cases, in which the objective functions are convex but not strongly convex. For example, a linear functional is convex but not strongly convex; so is a positive semidefinite quadratic functional. A question arises: given a convex function can we somehow "strongly convexify" it, so that we can apply the decomposition technique in a straightforward manner? The answer is yes. It can be done by using the proximal point method, but at the expense that we have to solve a sequence of problems instead of just one.

2.2 The Proximal Point Method

The notion of the proximal point was introduced by Moreau [1965]. A convergence proof of the proximal point method was given by Martinet [1970, 1972], Rockafellar [1976 a, b] and Brézis and Lions [1978] gen-

eralized the results and gave the rate of convergence in two slightly different versions. Rockafellar applied it to various types of mathematical programming problems. (The proximal point is also called the resolvent method.)

We consider a general convex programming problem

$$\min f(x) \qquad (2.7)$$
$$x \in C$$

where f is a convex function defined on $\mathbb{R}^n$, having values in $\mathbb{R}$ and C is a nonempty closed convex set in $\mathbb{R}^n$. Let ψ_C be the indicator function of the set C, i.e.,

$$\psi_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{otherwise .} \end{cases}$$

Define $F(x) := f(x) + \psi_C(x)$; then F is a proper closed convex function and (2.7) can be rewritten as

$$\min_{x \in \mathbb{R}^n} F(x)$$

The subdifferential $\partial F(x)$ of F at x is defined by

$$\partial F(x) := \{t \in \mathbb{R}^n \mid F(z) \geq F(x) + \langle t, z-x \rangle \ \forall z \in \mathbb{R}^n\} .$$

It is easy to see from the definition of ∂F that x^* is an optimal solution of (2.7) iff $0 \in \partial F(x^*)$. Therefore, the minimization problem (2.7) can be solved by finding a solution of a generalized equation $0 \in \partial F(x)$. We now study the problem of finding a solution for the generalized equation

$$0 \in T(x)$$

where T is a set-valued mapping. Let $D(T)$ be the domain of T, i.e. $D(T)$ is the set of x such that $T(x) \neq \phi$. T is said to be a monotone operator if

$$\langle x-x', y-y' \rangle \geq 0 \quad \forall y \in T(x), \forall y' \in T(x'), \quad \forall x, x' \in D(T) .$$

T is said to be a maximal monotone operator if it is monotone and its graph

$$G(T) := \{(x,y) \mid y \in T(x)\}$$

is not properly contained in the graph of any other monotone operator.

If F is a proper closed convex function, then the subdifferential ∂F is a maximal monotone operator (see [Brézis 1973]). Given a positive number λ, the resolvent J_λ of T is defined by

$$J_\lambda(y) := (I+\lambda T)^{-1}(y) .$$

J_λ is a single-valued function and it is a contraction, i.e.,

$$\|J_\lambda(y_1)-J_\lambda(y_2)\| \leq \|y_1-y_2\| \quad \text{for all } y_1 \text{ and } y_2 .$$

(see also [Brézis 1973]). We have an important relation

$$0 \in T(x) \text{ if and only if } x = J_\lambda(x) . \tag{2.8}$$

The minimization problem (2.7) has thus been transformed to the problem of finding a fixed point of the resolvent J_λ of the subdifferential $\partial F = \partial(f+\psi_C)$.

The proximal point algorithm generates for any starting point

x^0 a sequence $\{x^k\}$ obtained by the relation

$$x^{k+1} = J_{\lambda_k}(x^k) \tag{2.9}$$

where $\{\lambda_k\}$ is a sequence of positive numbers with $\lambda_k \geq \lambda > 0 \ \forall\ k$. In the case $T = \partial F$, x^{k+1} is the optimal solution of

$$\min_x F(x) + \frac{1}{2\lambda_k} \|x-x^k\|^2 ,$$

which is equivalent to

$$\min_{x \in C} f(x) + \frac{1}{2\lambda_k} \|x-x^k\|^2 . \tag{2.10}$$

In practice, it is impossible to obtain the true optimal solution of (2.10), so we would like to be able to choose x^{k+1} as a point near the optimal solution. We shall use the following approximation criterion

$$\|x^{k+1} - J_{\lambda_k}(x^k)\| \leq \varepsilon_k \tag{2.11}$$

where $\{\varepsilon_k\}$ is a sequence of positive numbers such that $\sum_{k=1}^{\infty} \varepsilon_k < \infty$.

Theorem 23 ([Rockafellar, 1976a])

Let $\{x^k\}$ be any sequence generated by the proximal point algorithm under the criterion (2.11). Suppose $\{x^k\}$ is bounded. Then $\{x^k\}$ converges to a point x^* satisfying $0 \in T(x^*)$ and

$$\lim_{k\to\infty} \|x^{k+1} - x^k\| = 0 .$$

Remark

(i) The necessary and sufficient condition for the boundedness of $\{x^k\}$ is the existence of a solution for $0 \in T(x)$ ([Rockafellar, 1976a) (i.e. $\{x^k\}$ is bounded if and only if (2.7) has optimal solutions).

(ii) Under the condition that T^{-1} is Lipschitz continuous at 0, the sequence $\{x^k\}$ converges to x^* linearly. In addition, if $\lambda_k \uparrow \infty$, the convergence is superlinear.

If there exists $\bar{x}$ such that $0 \in \text{int } T(\bar{x})$, then, with the exact form $x^{k+1} = J_{\lambda_k} x^k$, the convergence is finite. This is also true in the case of linear programming problems.

Application of the proximal point algorithm to linear programming problem

Consider a linear programming problem

$$\begin{aligned} &\min \ \langle c,x \rangle \\ &\text{subject to} \quad Ax = b \\ &\qquad\qquad x \geq 0 . \end{aligned}$$

Suppose the problem has an optimal solution. Let λ be a positive number. We have

$$\langle c,x \rangle + \frac{1}{2\lambda} \|x-x^k\|^2 = \frac{1}{2\lambda} \|x-(x^k-\lambda c)\|^2 - \frac{\lambda}{2} \|c\|^2 + \langle c,x^k \rangle .$$

Thus the problem

$$\begin{aligned} &\min \ \langle c,x \rangle + \frac{1}{2\lambda} \|x-x^k\|^2 \\ &\text{subject to} \quad Ax = b \\ &\qquad\qquad x \geq 0 \end{aligned}$$

is equivalent to

$$\min \quad \|x - (x^k - \lambda c)\|^2$$

$$\text{subject to} \quad Ax = b$$

$$x \geq 0 ,$$

so x^{k+1} is the projection of $x^k - \lambda c$ onto the feasible region. The proximal point algorithm in this case is the same as the gradient projection method of Levitin and Polyak [1966] or as one of the methods of feasible directions of Zoutendijk [1976] (but not as the gradient projection method of Rosen [1960], since x^k and x^{k+1} do not need to be on the same face of the feasible polyhedron, as in Rosen's method).

The figure below shows the sequence $\{x^k\}$ in an example.

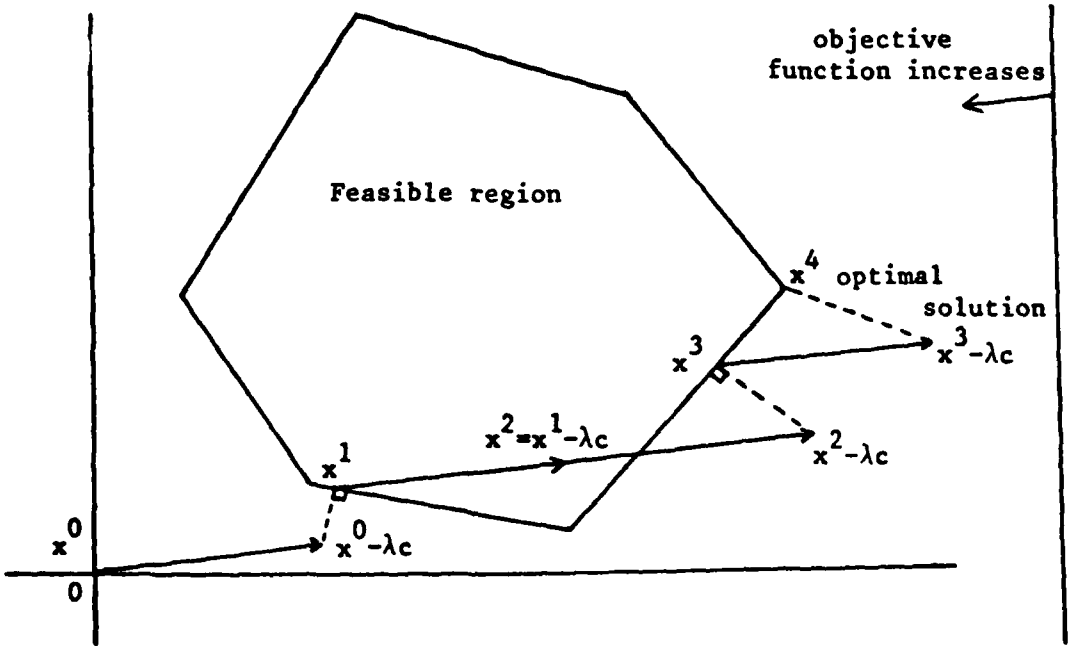


Figure 2.2 An example of the proximal point algorithm.

2.3 General algorithm

The decomposition technique and the proximal point method will be combined to give a new algorithm for problems having the form (2.1). For convenience, we rewrite (2.1) here.

$$\min \quad \sum_{i=1}^{m} f_i(x_i) \tag{2.1}$$

$$\text{subject to} \quad \sum_{i=1}^{m} A_i x_i = a \ .$$

Let $\mathcal{D}_i := \text{dom } f_i$, $f(x) := \sum_{i=1}^{m} f_i(x_i)$, and

$$C := \{x = (x_1, x_2, \ldots, x_m) \mid A_1 x_1 + A_2 x_2 + \ldots + A_m x_m = a\} \ .$$

Suppose (2.1) is feasible, i.e. $C \cap \prod_{i=1}^{m} \mathcal{D}_i \neq \phi$. We apply the proximal point algorithm to solve (2.1). First we choose a sequence of positive numbers $\{\lambda_k\}$, which is bounded away from zero; and a starting point $x^0 = (x_1^0, x_2^0, \ldots, x_m^0)$, which is not necessarily feasible. Suppose we have generated k points $x^1, x^2, \ldots, x^k$; then x^{k+1} will be the unique solution of the following problem:

$$\min \quad \sum_{i=1}^{m} \left(f_i(x_i) + \frac{1}{2\lambda_k} \|x_i - x_i^k\|^2\right) \tag{2.12}$$

$$\text{subject to} \quad \sum_{i=1}^{m} A_i x_i = a \ .$$

If $\|x^{k+1} - x^k\| < \varepsilon$, then x^{k+1} is an optimal solution for the original problem (2.1). Otherwise, replace x^k by x^{k+1} in (2.12) and repeat the procedure.

The problem (2.12) above has a strongly convex objective function, so we can apply the decomposition technique in a straightforward manner to solve it. That means transforming it to the dual problem (2.5) which is

$$\sup_{y} g(y)$$

where the function g, defined by (2.4), in this case is

$$g(y) = \langle a,y \rangle + \sum_{i=1}^{m} \min_{x_i} \left(f_i(x_i) + \frac{1}{2\lambda_k} \|x_i - x_i^k\|^2 - \langle A_i^T y, x_i \rangle \right) . \quad (2.13)$$

The derivative $g'(y)$ is given by

$$g'(y) = a - \sum_{i=1}^{m} A_i \bar{x}_i \quad (2.14)$$

where $\bar{x}_i$ is the optimal solution of

$$\min_{x_i} \left(f_i(x_i) + \frac{1}{2\lambda_k} \|x_i - x_i^k\|^2 - \langle A_i^T y, x_i \rangle \right) \quad (2.15)$$

for $i = 1,2,\ldots,m$.

Note that (i) $g(y)$ is finite for every y, so that the dual problem (2.5) is an unconstrained maximization problem. (ii) Once we compute m subproblems (2.15) to obtain the value of $g(y)$, we have the derivative $g'(y)$ almost for free. Hence we can use any gradient-type algorithm for unconstrained maximization problems to solve (2.5). In a schematic way, we have

Algorithm

Step 0 Choose a starting point $x^0 = (x_1^0, x_2^0, \ldots, x_m^0)$, set $k = 0$.

Step 1 1.0 Choose a point y_0; set $j = 0$.

1.1 Corresponding to y_j solve m minimization problems (2.15). Let $\bar{x}_i$ be the optimal solutions.

1.2 Compute $g(y_j)$ and $g'(y_j)$ as in (2.13) and (2.14).

1.3 If $\|g'(y_j)\| < \varepsilon$, go to step 2; otherwise, use some gradient-type algorithm to find a new point y_{j+1}, set $j = j + 1$, go back to step 1.1.

Step 2 If $\|\bar{x} - x^k\| < \varepsilon$, $\bar{x}$ is an optimal solution; otherwise, set $x^{k+1} = \bar{x}$ and $k = k + 1$, then go back to step 1.

Comments

(i) The fact that the starting point x^0 need not be feasible is very useful. In applications, usually by investigating the real situation from which the problem arises, we may have a guess which is infeasible but close to the optimal solution.

(ii) Even if x^0 is infeasible, the points $x^1, x^2, \ldots$ are feasible and $f(x^1) \geq f(x^2) \geq \ldots$. Hence, we have upper bounds which get better and better at each iteration.

(iii) We assumed that the problem (2.1) is feasible, so that

$$F(x) = f(x) + \psi_C(x)$$

is a proper closed convex function and ∂F is a maximal monotone operator. Consequently, the proximal point method goes through. Suppose we do not know whether the problem is feasible or not; can we apply the algorithm to solve the problem? If the problem turns out to be infeasible, what will tell us that?

The algorithm can be used without knowing the feasibility of the problem in advance. If the problem is infeasible then the problem (2.5) is unbounded. This is to say that if the primal problem is infeasible then the feasible dual is unbounded.

The general algorithm will be used to solve block-angular linear programming problems in the following section. We also use the general algorithm to solve nonlinear, convex structural engineering problems. The results are reported in [Kaneko and Ha, 1980] .

3.1 The Problem

A block angular linear programming problem is a linear programming problem having the form

$$\begin{aligned} \min \quad & \langle c_1, x_1 \rangle + \langle c_2, x_2 \rangle + \ldots + \langle c_m, x_m \rangle \\ \text{subject to} \quad & \\ & D_1 x_1 = d_1 \\ & D_2 x_2 = d_2 \\ & \vdots \\ & D_m x_m = d_m \\ & A_1 x_1 + A_2 x_2 + \ldots + A_m x_m = a \\ & x_1 \geq 0, \ldots, x_m \geq 0 \end{aligned} \tag{3.1}$$

where $x_i \in R^{n_i}$, $c_i \in R^{n_i}$, $d_i \in R^{m_i}$, $a \in R^{m_0}$, A_i is an $\ell \times n_i$ matrix, and D_i is an $m_i \times n_i$ matrix for $i = 1,2,\ldots,m$.

An example of a real world problem that has the special form (3.1) is the problem of a multidivisional organization. Each division operates with considerable autonomy; it has its own internal resources for production, e.g., labor and machines. That accounts for the constraints $D_i x_i = d_i$ $i = 1,\ldots,m$. The divisions are coupled by the fact that there are shared resources which all of the divisions use, for example a raw material of limited availability. That gives rise

to the coupling constraints

$$\sum_{i=1}^{m} A_i x_i = a \ .$$

Approaches to solving (3.1) can be roughly divided into two categories: improvements of the simplex method and decomposition techniques. In the simplex method the main computational difficulties are the updating of the inverse of the basis. Because the problem (3.1) has a special structure, there are several ways to reduce the computational effort and/or the storage requirements of each simplex iteration. The principal improvements are generalized upper bounding techniques ([Dantzig and Van Slyke, 1967]), basis factorization ([Winkler, 1974]), and LU decomposition ([Bartels and Golub, 1969] and [Forrest and Tomlin, 1972]). The well-known paper of Dantzig and Wolfe [1960] was the first paper to use a decomposition technique to solve (3.1). The idea of the Dantzig-Wolfe decomposition is appealing but computational experiences are erratic ([Lasdon, 1978] and [Adler and Ülkücü, 1973]). Recently there have been several attempts to improve the computational aspects of the Dantzig-Wolfe decomposition, such as the work of Ten Kate [1972], or the boxstep algorithm of Marsten et al. [1975]. Up to the present, there are no reports on how those algorithms compare with each other.

3.2 The Algorithm

We are going to apply the general algorithm of Section 2 to solve (3.1). First we need to rewrite (3.1) in the form (2.1).

Define $\mathcal{D}_i := \{x_i \in \mathbb{R}^{n_i} \mid D_i x_i = d,\ x_i \geq 0\}$ for $i = 1,2,\ldots,m$. Suppose $\mathcal{D}_i \neq \phi$ for all i (if there exists an i such that $\mathcal{D}_i = \phi$, then the problem (3.1) is infeasible). Let $f_i(x_i) := \langle c_i, x_i \rangle + \psi_{\mathcal{D}_i}(x_i)$ for $i = 1,2,\ldots m$ then f_i are proper closed convex functions. The problem (3.1) now has the form

$$\min \sum_{i=1}^{m} f_k(x_i)$$

$$\text{subject to} \sum_{i=1}^{m} A_i x_i = a ,$$

which is the same form as (2.1). Therefore, we can apply the general algorithms to solve it. The subproblems (2.15), in this case, are quadratic programming problems

$$\begin{aligned} &\min_{x_i} \left(f_i(x_i) + \frac{1}{2\lambda_k} \|x_i - x_i^k\|^2 - \langle A_i^T y, x_i \rangle\right) \\ &= \min_{x_i} \left(\langle c_i, x_i \rangle + \psi_{\mathcal{D}_i}(x_i) + \frac{1}{2\lambda_k} \|x_i - x_i^k\|^2 - \langle A_i^T y, x_i \rangle\right) \\ &= \min_{x_i} \left(\frac{1}{2\lambda_k} \|x_i - x_i^k\|^2 + \langle c_i - A_i^T y, x_i \rangle\right) \end{aligned} \tag{3.2}_i$$

$$\text{subject to} \quad D_i x_i = d_i$$

$$x_i \geq 0 .$$

The function g, defined by (2.4), can be proved to be a piecewise quadratic function. We shall prove and use that fact later. For the moment, we use a general nonlinear algorithm to solve the dual problem (2.5) which is

$$\sup_{y \in \mathbb{R}^{\ell}} g(y)$$

with

$$g(y) = \langle a,y \rangle - \sum_{i=1}^{m} \min_{x_i \in \mathcal{D}_i} \left(\frac{1}{2\lambda_k} \|x_i - x_i^k\|^2 + \langle c_i - A_i^T y, x_i \rangle \right) .$$

The overall procedure can be summarized as follows:

Algorithm I

Step 0 Choose a starting point x^0 . Set $k = 0$.

Step 1

1.0 Choose a point y_0 . Set $j = 0$.

1.1 For given j and y_j solve $(3.2)_j$ for $j = 1,2,\ldots,m$. Let $\bar{x}_i$ be the optimal solution and $\bar{z}_i$ be the optimal objective value.

1.2 Compute $g(y_j) = \langle a,y_j \rangle - \sum_{i=1}^{m} \bar{z}_i$

$$g'(y_j) = a - \sum_{i=1}^{m} A_i \bar{x}_i$$

1.3 If y_j is a maximizer of g, go to Step 2; otherwise, use some gradient-type algorithm to find a new point y_{j+1}, set $j = j + 1$ and go back to Step 1.1.

Step 2 If $\|\bar{x} - x^k\| < \varepsilon$, accept $\bar{x}$ as (nearly) an optimal solution for (3.1); otherwise, set $x^{k+1} = \bar{x}$ and $k = k + 1$, then go back to Step 1.

Notes

(i) The iteration on x^k will be called the outer loop and that on y_j will be called the inner loop.

(ii) As mentioned in Section 2, if the problem (3.1) has optimal solutions then the outer loop is a finite process, i.e. x^k is an optimal solution for some k.

(iii) The fact that the starting point x^0 can be any point is clearly an advantage of our algorithm over algorithms based on the simplex method. For the simplex-type methods, the starting point needs to be not only a feasible point but also an extreme point of the feasible region, and we also need to know the inverse of the corresponding basis.

For a linear programming problem, the dual variables play an important role. They have economic interpretations and they are useful in sensitivity analysis. If we solve a linear programming problem by our algorithm, can we obtain the optimal dual variables?. The answer is yes. We have the following proposition.

Proposition 3.1

Consider a linear programming problem

$$\begin{aligned} &\min \ \langle c,w \rangle \\ &\text{subject to} \ \ Aw = b \qquad (3.3) \\ &\qquad\quad w \geq 0 . \end{aligned}$$

Suppose it has an optimal solution $\bar{w}$. Then the quadratic programming problem

$$\begin{aligned} \min \quad & \langle c,w\rangle + \frac{1}{2\lambda}\|w-\bar{w}\|^2 \\ \text{subject to} \quad & Aw = b \\ & w \geq 0 \end{aligned}$$

has the Karush-Kuhn-Tucker point ([Mangasarian, 1969]) (w^*,v^*), where $w^* = \bar{w}$ and v^* is an optimal dual variable for (3.3).

Proof

The proof of the identity $w^* = \bar{w}$ is given by relation (2.8) of Section 2. (3.4) is equivalent to

$$\begin{aligned} \min \quad & \langle c - \frac{1}{\lambda}\bar{w},w\rangle + \frac{1}{2\lambda}\langle w,w\rangle \\ \text{subject to} \quad & Aw = b \\ & w \geq 0 . \end{aligned}$$

Hence (w^*,v^*) has to satisfy the following relations

$$c - \frac{1}{\lambda}\bar{w} + \frac{1}{\lambda}w^* - A^Tv^* \geq 0$$

$$w^* \geq 0 \quad Aw^* = b$$

$$\langle w^*,(c - \frac{1}{\lambda}\bar{w} + \frac{1}{\lambda}w^* - A^Tv^*)\rangle = 0$$

Since $w^* = \bar{w}$, those relations reduce to

$$c - A^Tv^* \geq 0$$

$$\bar{w} \geq 0 \quad A\bar{w} = b$$

$$\langle \bar{w},\ c - A^Tv^*\rangle = 0$$

which are the complementary slackness conditions for (3.3). Thus v^* is an optimal dual variable for (3.3). □

3.3 Implementation and Computational Results

We now consider the details of how we solved the dual problem (2.5) and the quadratic programming problems $(3.2)_i$. Recall that $g(y)$ is a Lipschitz continuously differentiable function, so to solve (2.5) we can use any gradient-type unconstrained minimization algorithm. We think that the best current algorithm is that of Broyden-Fletcher-Golfard-Shanno (BFGS). We used the BFGS package of the Harwell Subroutine Library available at the Madison Academic Computing Center and known as VA13A ([MACC, 1976]).

The core of our algorithm is the solution of small quadratic programming problems $(3.2)_i$; the number of quadratic programming problems solved in a problem may be several hundreds. For that reason, we need an efficient quadratic programming algorithm which can take the optimal solutions of the problems of the previous iteration as the starting point for the problems of the current iteration. Furthermore, we want to exploit the fact that the matrices in the objective functions are the identity matrices times a constant. First, we transform problems $(3.2)_i$ into quadratic programming problems having only nonnegativity constraints. Problems $(3.2)_i$ have the following form, except for constant terms

$$\begin{aligned} \min \quad & \langle \bar{c}, u \rangle + \frac{1}{2\lambda} \langle u, u \rangle \\ \text{subject to} \quad & \bar{D}u = \bar{d} \\ & u \geq 0 \end{aligned} \tag{3.5}$$

where, for a fixed i, $\bar{D} = D_i$, $\bar{d} = d_i$, $u = x_i$, and $\bar{c} = c_i - A_i^T y - \frac{1}{\lambda} x_i^k$.

Using the orthogonal projector P onto the null space of $\bar{D}$ and the ordinary dual of a quadratic programming problem, we can show that (3.5) is equivalent to

$$\max_{v \geq 0} \; -\frac{\lambda}{2} \langle v, \underline{P}v \rangle + \langle \lambda P\bar{c} - \hat{u}, v \rangle \qquad (3.6)$$

where $\hat{u}$ is an arbitrary solution for $\bar{D}u = \bar{d}$.

The orthogonal projector P and the solution $\hat{u}$ can be computed easily by using the Moore-Penrose generalized inverse and the QR decomposition of $\bar{D}$ (for more detail see [Ha, 1980]). Note that, for a fixed i, $\bar{D}$ and $\bar{d}$ do not change at all, so we need to compute P and $\hat{u}$ only once.

There are several quadratic programming algorithms that can be used to solve (3.6). There are two packages available at the Madison Academic Computing Center of the University of Wisconsin-Madison, namely QUADPR and LCPL, but with these packages we cannot use the optimal solutions of the previous iteration as the starting points for the current iteration. The Best-Ritter algorithm ([Best and Ritter, 1976]) allows us to do that, so we use it to solve (3.6).

We wrote a computer program to test the algorithm, using Fortran V on the UNIVAC 1110 of the Madison Academic Computing Center of the University of Wisconsin-Madison. In the program we set the stopping criteria as follows.

(i) Outer loop (Step 2). We set $\varepsilon_1 = 10^{-4}$. If

$$\|x^{k+1}-x^k\| \leq \varepsilon$$

or

$$\|x^{k+1}-x^k\| \leq \varepsilon_1 \|x^k\|$$

then x^{k+1} is considered to be an optimal solution for the problem.

(ii) Inner loop (Step 1.3). We set $\varepsilon_2 = 10^{-5}$. The stopping criterion in this case is that of VA13A. That means a solution is accepted if a relative change of size ε_2 in the components of y does not reduce the objective value.

We generated test problems by predetermining the size of the problem, the size of blocks and the number of coupling constraints, and then generating randomly data of the problem. The matrices A_i and D_i are 90% dense. Each entry of those matrices is a pseudo-random number in the range [-50, 50] (obtained by the random number routines of the Madison Academic Computing Center [MACC, 1978]). The cost coefficients are pseudo-random numbers in the range [-10, 10]. To be sure that the problem is feasible we randomly generated a sequence of integers in between 0 and 5 (considered as a feasible point) and then multiplied them with D_i and A_i to get the coefficients of the right hand side. The numbers of variables of test problems and other information are shown in Table 3.1.

Problem	Problem Size	Number of Blocks	Number of Coupling Constraints
I	50 × 100	10	5
II	70 × 95	3	10
III	100 × 200	20	5
IV	500 × 700	20	10

Table 3.1 Test Problems Statistics

We compared our algorithm with two linear programming packages available at the Madison Academic Computing Center of the University of Wisconsin-Madison, namely SIMPLX and FMPS-LP ([MACC, 1977 and 1978b]). SIMPLX uses the two-phase, revised simplex method with the inverse of the basis matrix stored explicitly. FMPS-LP is a part of the UNIVAC Functional Mathematical Programming System. It uses variants of the revised simplex method with only nonzero elements of the constraint matrix stored explicitly, and the inverse stored in the product form. In all cases, our algorithm gave the same optimal solutions (up to five significant figures) as the linear programming packages. We kept λ unchanged from one iteration to the next of the outer loop, but we ran the problems with different λ. Computational results are shown in the following tables. CPU time includes time to collect all relocatable elements and to produce an executable absolute element, and time for input/output. It is measured in seconds. The starting point x^0 and the point y_0 were taken to be the origin.

λ	Problem			
	I	II	III	IV[1]
1	104.0	48.5	219.2	
10	23.7	68.8[2]	34.5	724.5
20	19.2	99.5	34.1	
30	15.1	80.0	27.4	
40	13.2	100.0	24.9	
50	12.6	117.5	47.7	
60	10.5	101.5	23.8	
70	13.3	126.0	24.5	
80	10.5	105.5	20.5	
90	10.2	170.8	43.5	
100	9.7	102.5	28.6	

Table 3.2 CPU Time (secs) of Test Problems

Compare to CPU time of the linear programming packages.

	I	II	III	IV
SIMPLX	32.5	82.3	257.8	(3)
FMPS-LP	6.5	14.7	24.8	3617.0(4)

Table 3.3 CPU Time (secs) of Linear Programming Packages

(1) Because of the size of the problem and of the limitation of our budget, we decided to run only one run with $\lambda = 10$.

(2) There is noise for $\lambda \geq 10$, i.e. it came near the optimal solution and then moved around that point erratically.

(3) SIMPLX cannot handle a problem of that size (500×700).

(4) The problem needed 5 runs to reach the optimal solution, so the CPU time included time for putting data and current tableaus on a file, and retrieving them. The actual time of solving the problem should be less than 3617.0 seconds, but certainly it is much longer than 724.5 seconds, the time that our algorithm took to solve the problem.

From these tables we have several observations.

(1) Comparing to FMPS-LP, our algorithm gets better when the size of the problem increases. For Problems I and II, none of the runs of our algorithm is faster than FMPS-LP; for Problem III (size 100×200), there are several λ with which our algorithm is faster than FMPS-LP; for Problem IV our algorithm is clearly

far superior to FMPS-LP. Certainly the regular SIMPLX is not comparable to our algorithm.

(2) Our algorithm is not very good in the case of Problem II, which has 3 big blocks (the sizes of blocks are 15×25, 20×30, and 25×40). That is expected. The core of our algorithm is solving quadratic programming problems (3.2) again and again. If we have, say 10 smaller subproblems instead of 3 subproblems of the above sizes, we would have a much better time, since the computational time tends to grow polynomially with respect to the size of the problem.

(3) The values of λ, which give the best computational times, vary with problems: they are 100, 1 and 80 for problems I, II and III respectively. In theory, the number of iterations of the outer loop decreases as λ increases (in fact we can prove that if we use a λ sufficiently big, we could reach an optimal solution in one iteration of the outer loop). But by taking λ too large, we may have numerical difficulties and may not be able to solve the problem at all. Fewer iterations of the outer loop does not mean less computational time as it can be seen on Table 3.2. The other drawback of a large λ is the loss of accuracy caused by numerical errors. A natural alternative seems to be: taking λ initially large then reducing it gradually. But that idea did not work well. We have run our test problems with different schemes of changing λ from one iteration to the next and we

found that the computational results are very erratic. That is understandable: since we have not been able to choose λ optimally for a given problem, we should not expect to know how to change λ iteratively to get a better computational time. We feel that the question of choosing and/or changing λ iteratively can only be answered after an extensive use of our algorithm. For the moment we suggest to use λ fixed with a value in the range from 10 to 50; if numerical difficulties are encountered reduce λ .

As mentioned earlier one of the advantages of our algorithm is that the starting point x^0 can be anywhere. Suppose we know approximately where the optimal solution should be; then the computation time should be better. We ran our test problems with the starting points taken to be the known optimal solutions, rounded to the nearest integer. The results, which are better as would be expected, are shown in the following table.

	I	II	III	
starting point $x^0 = 0$	23.7	68.8	34.5	($\lambda = 10$)
starting point x^0 near optimal solution	8.4	58.7	24.2	

Table 3.4 CPU Time of Test Problems with Different Starting Points

3.4 A Variant of Algorithm I (Method of feasible directions)

We know that, if the problem (3.1) has optimal solutions, then the proximal point algorithm (outer loop) will generate a finite sequence of feasible points $\{x^1, x^2, \ldots, x^k\}$ such that $f(x^{j+1}) \leq f(x^j)$ for $j = 1,2,\ldots,k-1$, and x^k is an optimal solution for (3.1). But if the problem is unbounded, then the sequence $\{x^k\}$ is infinite and $\|x^k\| \to \infty$ as $k \to \infty$. For practical purposes that is undesirable; we want to have a simpler criterion for the case of unboundedness. By exploiting the linearity of the problem we can modify the proximal point method so that it either obtains the optimal solution or detects the unboundedness in a finite number of iterations. We set λ fixed and replace Step 2 of Algorithm I by

Step 2' If $\|\bar{x} - x^k\| < \varepsilon$, accept $\bar{x}$ as an optimal solution for (3.1); otherwise, for $k = 1$ set $\alpha = 1$, for $k > 1$ compute

$$\alpha = \min \left\{ \frac{(x^k)_i}{(x^k - \bar{x})_i} \;\middle|\; \text{for indices } i \text{ such that } (x^k - x)_i > 0 \right\}. \quad (3.7)$$

If $(x^k - \bar{x})_i \leq 0$ for all i, the problem is unbounded. Otherwise, set $x^{k+1} = x^k + \alpha(\bar{x} - x^k)$ and $k = k + 1$, then go back to Step 1

Proposition 3.2

The modified algorithm with Step 2' above either detects the unboundedness of the problem or finds an optimal solution in a finite number of iterations of the outer loop.

Proof

For $k \geq 1$ x^k is feasible. By the interpretation of the proximal point method given in Chapter 2, $\bar{x}$ is the projection of $x^k - \lambda c$ onto the feasible region. If we consider λc as the cost coefficients instead of c, then $\bar{x}$ is the projection of the negative of the gradient at x^k onto the feasible region. α given by (3.7) is the maximum stepsize. Therefore, the outer loop procedure (Step 2') is a method of feasible directions applied to linear programming problems. The finiteness of the method is proved by Zoutendijk [1976].

We used the modified algorithm to solve our test Problems I and III; we had the following results.

λ	I	III
1	35.8(1)	81.4
10	25.9	45.9
20	20.0	30.5
30	16.5	33.8
40	12.3	29.9
50	14.2	104.3(2)
60	16.4	46.3
70	18.1	38.6
80	10.1	19.9
90	12.8	39.4
100	12.9	45.1

Table 3.5 CPU Time of Test Problems Using the Modified Algorithms

(1) For all λ, except $\lambda = 80$, it came near the optimal solution then moved around that point erratically.

(2) It came near the optimal solution in 22 seconds then went away again.

We also used the modified algorithm to solve an unbounded problem. The problem is of size 50×90, with 7 blocks and 10 coupling constraints.

Modified Algorithm With $\lambda =$							SIMPLX	FMPS-LP
10	20	30	40	50	60	70		
47.6	41.8	33.6	36.2	35.2	29.6	(1)	44.7	11.0

Table 3.6 CPU Time for An Unbounded Problem

(1) Numerical errors.

The modified algorithm has the advantage of detecting the unboundedness, but it is not very stable when getting near the optimal solution. The reason is that near optimum x^k and $\bar{x}$ are very close together, so numerical errors in x^k may give much bigger errors for the quotient $\frac{(x^k)_i}{(x^k - \bar{x})_i}$; consequently $x^{k+1} = x^k + \alpha(\bar{x} - x^k)$ may be a point farther from the optimal solution than x^k. Hence we recommend using Algorithm I for problems for which the existence of optimal solutions is known beforehand. Otherwise, we suggest using the modified algorithm, but switching back to Algorithm I when the distance between $\bar{x}$ and x^k is small.

3.5 A Specific Algorithm for the Dual Problem

Algorithm I is a straightforward application of the general algorithm in Section 2 to the block angular linear programming problem (3.1). We have not taken advantage of the linearity of the problem; in particular the algorithm used to solve the dual problem $\sup g(y)$ is suitable for general nonlinear optimization problems. Geoffrion [1970b] observes that the dual of a quadratic programming problem taken with respect to a subset of the constraints is a piecewise quadratic function. That is exactly our case and we are going to give a proof of the above observation. We consider the problem

$$\begin{aligned} \min \quad & \langle c,x\rangle + \tfrac{1}{2}\langle x,Cx\rangle \\ \text{subject to} \quad & Dx = d \\ & Ax = a \\ & x \geq 0 . \end{aligned} \tag{3.8}$$

Suppose C is symmetric and positive definite and at every point of the set $\{x \geq 0 \mid Dx = d\}$, the gradients of the active constraints are linearly independent. As before, for a given y we define $g(y)$ to be the objective value of the problem

$$\begin{aligned} \min \quad & \langle c,x\rangle + \tfrac{1}{2}\langle x,Cx\rangle + \langle y,a - Ax\rangle \\ \text{subject to} \quad & Dx = d \\ & x \geq 0 . \end{aligned} \tag{3.9}$$

Proposition 3.3

$g(y)$ is a piecewise quadratic function.

Proof

For a fixed y_0, let $x(y_0)$ be the optimal solution of (3.9) corresponding to y_0 and define the set of indices M by

$$M := \{i \mid x(y_0)_i = 0\} .$$

Consider the problem

$$\begin{aligned} \min \quad & \langle c,x\rangle + \tfrac{1}{2}\langle x,Cx\rangle + \langle y,a-Ax\rangle \\ \text{subject to} \quad & Dx = d \\ & x_i = 0 \quad \text{for} \quad i \in M . \end{aligned} \tag{3.10}$$

The problem above is feasible since $x(y_0)$ satisfies the constraints Let $x(M,y)$ be the optimal solution and $v(M,y)$ be the multipliers. Let J be the index set of $v(M,y)$ corresponding to the constraints

$$x_i = 0 \quad \text{for} \quad i \in M .$$

Let K be the matrix representing these constraints; that is K is a matrix of $|M|$ rows, with the k-th row having an entry of 1 in the column corresponding to the k-th element of M and zero in all other entries. Define

$$\hat{D} := \begin{bmatrix} D \\ K \end{bmatrix} \quad \text{and} \quad \hat{d} := \begin{bmatrix} d \\ 0 \end{bmatrix} ,$$

then the constraints of (3.10) can be rewritten as

$$\hat{D}x = \hat{d}$$

and the Karush-Kuhn-Tucker conditions for (3.10) are

$$\begin{cases} c + Cx - A^T y - \hat{D}^T v = 0 \\ \hat{D}x = \hat{d} \end{cases}$$

or

$$\begin{bmatrix} C & -\hat{D}^T \\ \hat{D} & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} A^T y - c \\ \hat{d} \end{bmatrix} . \tag{3.11}$$

Under our assumptions the matrix above is invertible; consequently $x(M,y)$ and $v(M,y)$ are affine functions of y.

$x(M,y)$ and $v(M,y)$, being the optimal solution and the Lagrange multipliers of (3.10), need not be the optimal solution and the Lagrange multipliers of (3.9). But if

$$x(M,y) \geq 0 \quad \text{and} \quad v_j(M,y) \geq 0 \quad \text{for} \quad j \in J ,$$

then, by setting all of the Lagrange multipliers corresponding to the constraints $x_i \geq 0$ $i \notin M$ equal to zero, we can verify easily that $x(M,y)$ and $v(M,y)$ satisfy the Karush-Kuhn-Tucker conditions for (3.9). Hence $x(M,y)$ in this case is the optimal solution for (3.9). Conversely, if $x(M,y)$ and $v(M,y)$ are the optimal solution and the Lagrange multipliers of (3.9), then certainly we have $x(M,y) \geq 0$ and $v_j(M,y) \geq 0$ for $j \in J$. The set

$$Q_M := \{y \mid x(M,y) \geq 0 \quad \text{and} \quad v_j(M,y) \geq 0 \quad \text{for} \quad j \in J\} \tag{3.12}$$

is a polyhedron. On each such polyhedron $g(y)$ is a quadratic function. Hence $g(y)$ is piecewise quadratic.

Using the above proposition we have another algorithm to solve (3.1).

Algorithm II

Step 0 Choose any point x^0 as the starting point, set $k = 0$.

Step 1

1.0 For given k and x^k, choose y_0 and set $j = 0$.

1.1 Solve

$$\min \quad \frac{1}{2\lambda_k}\|x_i - x_i^k\|^2 + \langle c_i - A_i^T y_j, x_i \rangle$$

$$\text{s.t.} \quad D_i x_i = d_i \qquad (3.13)$$

$$x_i \geq 0$$

for $i = 1,2,\ldots,m$.

Let $\bar{x}_i$ be the optimal solution and v_i be the corresponding multipliers. Let $\bar{x} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_m)$.

1.2 Determine the index set $M = \{h/(\bar{x})_h = 0\}$ and the corresponding polyhedron Q_M defined by (3.12). Solve the quadratic programming problem

$$\sup_{y \in Q_M} g(y) \quad . \qquad (3.14)$$

If it is unbounded, then the original problem (3.1) is infeasible. Otherwise, let $\bar{y}$ be the optimal solution.

1.3 If $\|g'(\bar{y})\| < \varepsilon$ go to Step 2. Otherwise compute

$$y_{j+1} = \bar{y} + \alpha g'(\bar{y})$$

where α is a real number such that $g(y_{j+1}) > g(\bar{y})$.

Set $j = j + 1$ and go back to Step 1.1.

Step 2 Compute $x(M,\bar{y})$ by (3.11). If $\|x(M,\bar{y}) - x^k\| < \varepsilon$, $x(M,\bar{y})$ is an optimal solution for (3.1). Otherwise set $x^{k+1} = x(M,\bar{y})$ and $k = k + 1$, then go back to Step 1.

Remark

The partition of the space of y into polyhedra is finite and in the process of finding the optimal solution of the dual problem a polyhedron is never repeated. This is true because in Step 1.3 we require $g(y_{j+1}) > g(\bar{y})$ and $\bar{y}$ is the maximizer of $g(y)$ on the polyhedron containing y_j, so that polyhedron is never repeated. Therefore, the inner loop is a finite procedure.

We now explain in detail how to determine the polyhedron Q_M in Step 1.2. Recall that Q_M is given by

$$Q_M = \{y | x(M,y) \geq 0 \quad \text{and} \quad v_j(M,y) \geq 0 \quad j \in J\}$$

where $x(M,y)$ and $v(M,y)$ solve the system of linear equations (3.11). We have

$$\begin{bmatrix} C & -\hat{D}^T \\ \hat{D} & 0 \end{bmatrix}^{-1} = \begin{bmatrix} C^{-1} - C^{-1}\hat{D}^T(\hat{D}C^{-1}\hat{D}^T)^{-1}\hat{D}C^{-1} & C^{-1}\hat{D}^T(\hat{D}C^{-1}\hat{D}^T)^{-1} \\ -(\hat{D}C^{-1}\hat{D}^T)^{-1}\hat{D}C^{-1} & (\hat{D}C^{-1}\hat{D}^T)^{-1} \end{bmatrix}$$

In our case $C = \lambda^{-1}I$ so we can simplify the above expression to obtain

$$\begin{bmatrix} I/\lambda - \hat{D}^T \\ \hat{D} \quad 0 \end{bmatrix}^{-1} = \begin{bmatrix} \lambda(I - \hat{D}^T(\hat{D}\hat{D}^T)^{-1}\hat{D}) & \hat{D}^T(\hat{D}\hat{D}^T)^{-1} \\ -(\hat{D}\hat{D}^T)^{-1}\hat{D} & \frac{1}{\lambda}(\hat{D}\hat{D}^T)^{-1} \end{bmatrix}$$

$$\begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} I/\lambda - \hat{D}^T \\ \hat{D} \quad 0 \end{bmatrix}^{-1} \begin{bmatrix} A^T y - c \\ \hat{d} \end{bmatrix}$$

$$x(M,y) = \lambda(I - \hat{D}^T(\hat{D}\hat{D}^T)^{-1}\hat{D})(A^T y - c) + \hat{D}^T(\hat{D}\hat{D}^T)^{-1}\hat{d}$$

$$v(M,y) = -(\hat{D}\hat{D}^T)^{-1}\hat{D}(A^T y - c) + \frac{1}{\lambda}(\hat{D}\hat{D}^T)^{-1}\hat{d}$$

Let $\hat{P}$ be the orthogonal projector onto the null space of $\hat{D}$ and $\hat{D}^+$ be the generalized inverse of $\hat{D}$. We know that

$$\hat{P} = I - \hat{D}^T(\hat{D}\hat{D}^T)^{-1}\hat{D}$$

and

$$\hat{D}^+ = \hat{D}^T(\hat{D}\hat{D}^T)^{-1} ,$$

so

$$x(M,y) = \lambda\hat{P}(A^T y - c) + \hat{D}^+\hat{d}$$

and (3.15)

$$v(M,y) = -(\hat{D}^+)^T(A^T y - c) + \frac{1}{\lambda}(\hat{D}\hat{D}^T)^{-1}\hat{d} .$$

Because $x(M,y)$ is the solution of (3.11), $x_i(M,y) = 0$ for all $i \in \mathcal{A}$. Hence Q_M can be expressed explicitly as

$$Q_M = \{y \mid (\lambda \hat{P}^T(A^T y - c) + \hat{D}^+ \hat{d})_i \geq 0, \quad i \notin M \text{ and}$$

$$(-(\hat{D}^+)^T(A^T y - c) + \frac{1}{\lambda}(\hat{D}\hat{D}^T)^{-1}\hat{d})_j \geq 0, \quad j \in J\}$$

Using the expression (3.15) of $x(M,y)$ we have an explicit expression for $g(y)$, $y \in Q_M$.

$$g(y) = \frac{1}{2\lambda}\langle x(M,y), x(M,y)\rangle + \langle c, x(M,y)\rangle + \langle y, a - Ax(M,y)\rangle$$

$$= -\frac{\lambda}{2}\langle \hat{P}A^T y, \hat{P}A^T y\rangle + \langle y, a + \lambda A\hat{P}c - A\hat{D}^+\hat{d}\rangle$$

$$+ \langle c, \hat{D}^+\hat{d} - \lambda\hat{P}c\rangle + \frac{1}{2\lambda}\langle \hat{D}^+\hat{d} - \lambda\hat{P}c, \hat{D}^+\hat{d} - \lambda\hat{P}c\rangle .$$

After deleting constants in the objective function, we see that the problem (3.14) is equivalent to

$$\sup \quad -\frac{\lambda}{2}\langle \hat{P}A^T y, \hat{P}A^T y\rangle + \langle y, a + \lambda A\hat{P}c - A\hat{D}^+\hat{d}\rangle$$

$$\text{subject to} \quad (\lambda\hat{P}(A^T y - c) + \hat{D}^+\hat{d})_i \geq 0 \quad i \in M$$

$$(-(\hat{D}^+)^T(A^T y - c) + \frac{1}{\lambda}(\hat{D}\hat{D}^T)^{-1}\hat{d})_j \geq 0 \quad j \in J .$$

In our original problem (3.1), the matrix D has the block angular form

$$D = \begin{bmatrix} D_1 & & & 0 \\ & D_2 & & \\ & & \ddots & \\ 0 & & & D_m \end{bmatrix} .$$

The matrix K representing constraints $x_i = 0$ also has the same

form. Rearranging the rows of $\hat{D}$, $\hat{D}$ has the form

$$\hat{D} = \begin{bmatrix} \begin{matrix} D_1 \\ K_1 \end{matrix} & & \bigcirc \\ & \begin{matrix} D_2 \\ K_2 \end{matrix} & \ddots \\ \bigcirc & & \begin{matrix} D_m \\ K_m \end{matrix} \end{bmatrix}$$

Let $\hat{D}_i = \begin{bmatrix} D_i \\ K_i \end{bmatrix}$; then

$$\hat{D} = \begin{bmatrix} \hat{D}_1 & & \bigcirc \\ & \hat{D}_2 & \ddots \\ \bigcirc & & \hat{D}_m \end{bmatrix} .$$

It is easy to see that

$$\hat{D}^+ = \begin{bmatrix} \hat{D}_1^T(\hat{D}_1\hat{D}_1^T)^{-1} & & \bigcirc \\ & \hat{D}_2^T(\hat{D}_2\hat{D}_2^T)^{-1} & \ddots \\ \bigcirc & & \hat{D}_m^T(\hat{D}_m\hat{D}_m^T)^{-1} \end{bmatrix}$$

and

$$\hat{P} = \begin{bmatrix} \hat{P}_2 & & \bigcirc \\ & \hat{P}_2 & \ddots \\ \bigcirc & & \hat{P}_m \end{bmatrix}$$

where $\hat{P}_i = I - \hat{D}_i^T(\hat{D}_i\hat{D}_i^T)^{-1}\hat{D}_i$.

Hence the block diagonal structure is preserved when we compute $\hat{D}^+$ and $\hat{P}$.

In updating y from y_j to y_{j+1}, we go from one polyhedron to another and we have to update $\hat{D}^+$ and $\hat{P}$. Usually the set of active constraints changes by just a few indices; consequently, we have to update only a few small matrices. By using the QR composition of $D_1, D_2, \ldots D_m$ that we already have, the updating of $\hat{D}^+$ and $\hat{P}$ is much more efficient ([Gill et al., 1974]).

We wrote a computer program to test the Algorithm II; again we used our test Problems I and III. We had the following results

λ	I	III
10	9.0	19.0
20	5.8	17.3
30	5.0	17.4
40	4.9	16.7
50	4.2	16.3
60	4.3	15.1
70	4.2	15.8
80	4.3	15.0
90	3.9	14.4
100	4.0	(1)

Table 3.7 CPU Time Using the Algorithm II

(1) Numerical errors.

We used the new system at MACC, the UNIVAC 1100/80, so these numbers are not comparable to the numbers of the Tables 3.2 and 3.3 which we obtained by using the old system, the UNIVAC 1110. The new system is approximately 2.5 times faster than the old one, so to compare the algorithms we need to multiply the numbers of Table 3.7 by a factor of 2.5. Then we can see that, for the Problem I, the Algorithm II is a little bit better then the Algorithm I, but, for the Problem III, the Algorithm II is not as good as the Algorithm I.

We think that the Algorithm II can be improved computationally. For the moment, in finding the maximizer for the piecewise quadratic function $g(y)$ we have not been able to use the optimal solution for $g(y)$ in one piece (polyhedron (3.12)) as a starting point for the problem in the next piece (problem 3.14)) (because, by choosing y_{j+1} as in Step 1.3, we are not sure that the polyhedron containing y_{j+1} is adjacent to that of y_j). Each time we solved the problem (3.14) we had to start from scratch, i.e., we had to use the simplex algorithm to find the starting point. That really slowed down the whole process.

ACKNOWLEDGEMENTS

This research is a part of the author's doctoral thesis submitted to the Department of Industrial Engineering at the University of Wisconsin-Madison under the supervision of Professor Stephen M. Robinson. I would like to express my sincere gratitude to Professor Stephen M. Robinson for his advice and support.

References

Adler, I. and Ülkücü (1973), :On the number of iteration in Dantzig-Wolfe decomposition algorithm," in Decomposition of Large-Scale Problems, Himmelblau (ed.), North Holland.

Bartels, R. H. and Golub, G. H. (1969), "The simplex method of linear programming using LU decomposition,," Communications ACM, Vol. 12, pp. 266-268 and 275-278.

Best, M. J. and Ritter, K. (1976), "An effective algorithm for quadratic minimization problems," Math. Res. Center, University of Wisconsin-Madison, Tech. Rep. #1691.

Brézis, H. (1973), Opérateurs Maximaux Monotones, North-Holland.

Brézis, H. and Lions, P. L. (1978), "Produits infini de Résolvantes," Israel J. Math., Vol. 12, pp. 329-345.

Dantzig, G. B. and Van Slyke, R. M. (1967), "Generalized upper bounded techniques for linear programming," J. Comp. System Sci., Vol. 1, pp. 213-226.

Dantzig, G. B. and Wolfe, P. (1960), "Decomposition principle for linear programs," Oper. Res., Vol. 8, No. 1, pp. 101-111.

Forrest, J. J. H. and Tomlin, J. A. (1972), "Updated triangular factors of the basis to maintain sparsity in the product form of simplex method," Math. Prog., VOl. 2, pp. 263-278.

Geoffrion, A. M. (1970), "Primal resource-directive approaches for optimizing nonlinear decomposable systems," Oper. Res., Vol. 18, No. 3, pp. 375-403.

Gill, P. E. et. al. (1974), "Methods for modifying matrix factorizations," Math. Comp., Vol. 28, pp. 505-535.

Ha, C. D. (1980), Decomposition methods for structured convex programming, Ph. D. Dissertation, Industrial Engineering Dept., Univ. Wisconsin-Madison.

Kaneko, I. and Ha, C. D. (1980), "A Decomposition procedure for large scale optimal plastic design problems," Math. Research Center, Univ. Wisconsin-Madison, Tech. Rep. #2075.

Lasdon, L. S. (1978), "Large-scale programming," in Handbook of Operations Research, Moder and Elmaghraby (eds), Van-Nostrand.

Levitin, E. S. and Polyak, B. T. (1966), "Constrained minimization methods," U.S.S.R. Comp. Math. and Math. Physics, Vol. 6, No. 5, pp. 1-50.

Madison Academic Computing Center (MACC, 1976), Nonlinear programming routines, reference manual.

Madison Academic Computing Center (MACC, 1977), Linear programming routines, reference manual.

Madison Academic Computing Center (MACC, 1978a), Random number routines, reference manual.

Madison Academic Computing Center (MACC, 1978b), FMPS-LP, reference manual.

Mangasarian, O. L. (1969), Nonlinear Programming, McGraw-Hill.

Marsten, R. E., Hogan W. W. and Blankenship, J. W. (1975), "The box-step method for large-scale optimization," Oper. Res., Vol. 23, No. 3.

Martinet, B. (1970), "Regularisation d'inéquations variationnelles par approximations successives," Rev. Fr. Inf. Rech. Oper., R. 3, pp. 154-159.

Martinet, B. (1972), "Détermination approchée d'un point fixe d'une application pseudo-contractante," C. R. Acad. Sci. Paris, Tome 274, serie A-163.

Morreau, J. J. (1965), "Proximité et dualité dans un espace Hilbertien," Bull. Soc. Math. Fr., Vol. 93, pp. 273-299.

Robinson, S. M. (1978), Private communication.

Rockafellar, R. T. (1970), Convex Analysis, Princeton University Press.

Rockafellar, R. T. (1976a), "Monotone operators and the proximal point algorithm," SIAM J. Control Opt., Vol. 14, No. 5.

Rockafellar, R. T. (1976b), "Augmented Lagrangians and applications of the proximal point algorithm in convex programming," Math. O. R., Vol. 1, No. 2.

Rosen, J. B. (1960), "The gradient projection method for nonlinear programming, part I: linear constraints," J. SIAM, Vol. 8, pp. 181-217.

Shapiro, J. F. (1978), "Nondifferentiable optimization and large scale linear programming," in Int. Symp. on Syst. Optim. Anal., Bensoussan and Lions (eds), Rocquencourt, France.

Ten Kate, A. (1972), "Decomposition of linear programs by direct distribution," Econometrica, Vol. 40, No. 5, pp. 353-363.

Winkler, C. (1974), "Basis factorization for block angular linear programs: unified theory of partitioning and decomposition using the simplex method," Tech. Rep. SOL 74019, Systems Optim. Lab., Stanford University.

Zoutendijk, G. (1976), Mathematical Programming Methods, North Holland.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: 2174 14 MRC-TSR-2174 / 2. GOVT ACCESSION NO.: AD-A099 362	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): A DECOMPOSITION METHOD AND ITS APPLICATION TO BLOCK ANGULAR LINEAR PROGRAMS	5. TYPE OF REPORT & PERIOD COVERED: Summary Report - no specific reporting period
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): 10 Cu Duong Ha	8. CONTRACT OR GRANT NUMBER(s): 15 DAAG29-80-C-0041
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Mathematics Research Center, University of Wisconsin, 610 Walnut Street, Madison, Wisconsin 53706	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 5- Operations Research
11. CONTROLLING OFFICE NAME AND ADDRESS: U. S. Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709	12. REPORT DATE: January 1981
	13. NUMBER OF PAGES: 51 (12) 35
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): 9 Technical Summary Rept.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Decomposition method, convex programming, large scale systems, linear programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper we propose and develop techniques for solving structured, large-scale convex programming problems. The procedure is a combination of a decomposition technique of Dantzig-Wolfe type and the proximal point method. The proximal point method is used to overcome the drawbacks of the decomposition technique.

The procedure is then used to solve block angular linear programming problems. By exploiting the linearity of the problem we have several variants of the procedure.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)


DAT
ILM